Semantics and Verification of Software

Lecture 18: Dataflow Analysis I (Introduction & Available Expressions Analysis)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH Aachen University

noll@cs.rwth-aachen.de

http://www-i2.informatik.rwth-aachen.de/i2/svsw10/

Summer Semester 2010

Outline

1. Preliminaries on Dataflow Analysis

2. An Example: Available Expressions Analysis

Dataflow Analysis: the Approach

- Traditional form of program analysis

Dataflow Analysis: the Approach

- Traditional form of program analysis
- Idea: describe how analysis information flows through program

Dataflow Analysis: the Approach

- Traditional form of program analysis
- Idea: describe how analysis information flows through program
- Distinctions:

  direction of flow: forward vs. backward analyses
  procedures: interprocedural vs. intraprocedural analyses
  quantification over paths: may (union) vs. must (intersection) analyses
  dependence on statement order: flow-sensitive vs. flow-insensitive analyses
  distinction of procedure calls: context-sensitive vs. context-insensitive analyses

Labelled Programs

- Goal: localization of analysis information

Labelled Programs

- Goal: localization of analysis information
- Dataflow information will be associated with
  - `skip` statements
  - assignments
  - tests in conditionals (`if`) and loops (`while`)

  These constructs will be called blocks.

Labelled Programs

- Goal: localization of analysis information
- Dataflow information will be associated with
  - `skip` statements
  - assignments
  - tests in conditionals (`if`) and loops (`while`)

  These constructs will be called blocks.
- Assume set of labels L with meta variable $l \in L$ (usually $L = \mathbb{N}$)

Labelled Programs

- Goal: localization of analysis information
- Dataflow information will be associated with
  - `skip` statements
  - assignments
  - tests in conditionals (`if`) and loops (`while`)

  These constructs will be called blocks.
- Assume set of labels L with meta variable $l \in L$ (usually $L = \mathbb{N}$)

Definition 18.1 (Labelled WHILE programs)

The syntax of labelled WHILE programs is defined by the following context-free grammar:

$$\begin{aligned}
a &::= z \mid x \mid a_1\texttt{+}a_2 \mid a_1\texttt{-}a_2 \mid a_1\texttt{*}a_2 \in AExp \\
b &::= t \mid a_1\texttt{=}a_2 \mid a_1\texttt{>}a_2 \mid \neg b \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \in BExp \\
c &::= [\texttt{skip}]^l \mid [x \texttt{ := } a]^l \mid c_1\texttt{;}c_2 \mid \\
&\qquad \texttt{if } [b]^l \texttt{ then } c_1 \texttt{ else } c_2 \mid \texttt{while } [b]^l \texttt{ do } c \in Cmd
\end{aligned}$$

Here all labels in a statement $c \in Cmd$ are assumed to be distinct.

A WHILE Program

Example 18.2

```
x := 6;
y := 7;
z := 0;
while x > 0 do
  x := x - 1;
  v := y;
  while v > 0 do
    v := v - 1;
    z := z + 1;
```

A WHILE Program with Labels

Example 18.2

$$
\begin{array}{l}
[\texttt{x := 6}]^1; \\
[\texttt{y := 7}]^2; \\
[\texttt{z := 0}]^3; \\
\texttt{while } [\texttt{x > 0}]^4 \texttt{ do} \\
\quad [\texttt{x := x - 1}]^5; \\
\quad [\texttt{v := y}]^6; \\
\quad \texttt{while } [\texttt{v > 0}]^7 \texttt{ do} \\
\qquad [\texttt{v := v - 1}]^8; \\
\qquad [\texttt{z := z + 1}]^9
\end{array}
$$

Representing Control Flow I

Every (labelled) statement has a single entry (given by the initial label) and generally multiple exits (given by the final labels):

Definition 18.3 (Initial and final labels)

The mapping $\mathsf{init} : Cmd \to L$ returns the initial label of a statement:

$$\begin{aligned}
\mathsf{init}([\texttt{skip}]^l) &:= l \\
\mathsf{init}([x \texttt{ := } a]^l) &:= l \\
\mathsf{init}(c_1 \texttt{;} c_2) &:= \mathsf{init}(c_1) \\
\mathsf{init}(\texttt{if } [b]^l \texttt{ then } c_1 \texttt{ else } c_2) &:= l \\
\mathsf{init}(\texttt{while } [b]^l \texttt{ do } c) &:= l
\end{aligned}$$

Representing Control Flow I

Every (labelled) statement has a single entry (given by the initial label) and generally multiple exits (given by the final labels):

Definition 18.3 (Initial and final labels)

The mapping $\mathsf{init} : \mathit{Cmd} \to L$ returns the initial label of a statement:

$$\begin{aligned}
\mathsf{init}([\mathtt{skip}]^l) &:= l \\
\mathsf{init}([x \texttt{ := } a]^l) &:= l \\
\mathsf{init}(c_1 \,;\, c_2) &:= \mathsf{init}(c_1) \\
\mathsf{init}(\mathtt{if}\ [b]^l\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2) &:= l \\
\mathsf{init}(\mathtt{while}\ [b]^l\ \mathtt{do}\ c) &:= l
\end{aligned}$$

The mapping $\mathsf{final} : \mathit{Cmd} \to 2^L$ returns the set of final labels of a statement:

$$\begin{aligned}
\mathsf{final}([\mathtt{skip}]^l) &:= \{l\} \\
\mathsf{final}([x \texttt{ := } a]^l) &:= \{l\} \\
\mathsf{final}(c_1 \,;\, c_2) &:= \mathsf{final}(c_2) \\
\mathsf{final}(\mathtt{if}\ [b]^l\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2) &:= \mathsf{final}(c_1) \cup \mathsf{final}(c_2) \\
\mathsf{final}(\mathtt{while}\ [b]^l\ \mathtt{do}\ c) &:= \{l\}
\end{aligned}$$

Representing Control Flow II

Definition 18.4 (Flow relation)

Given a statement $c \in Cmd$, the (control) flow relation $\mathsf{flow}(c) \subseteq L \times L$ is defined by

$$
\begin{aligned}
\mathsf{flow}([\mathtt{skip}]^l) &:= \emptyset \\
\mathsf{flow}([x \;\mathtt{:=}\; a]^l) &:= \emptyset \\
\mathsf{flow}(c_1\,\mathtt{;}\,c_2) &:= \mathsf{flow}(c_1) \cup \mathsf{flow}(c_2) \cup \\
&\ \{(l, \mathsf{init}(c_2)) \mid l \in \mathsf{final}(c_1)\} \\
\mathsf{flow}(\mathtt{if}\ [b]^l\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2) &:= \mathsf{flow}(c_1) \cup \mathsf{flow}(c_2) \cup \\
&\ \{(l, \mathsf{init}(c_1)), (l, \mathsf{init}(c_2))\} \\
\mathsf{flow}(\mathtt{while}\ [b]^l\ \mathtt{do}\ c) &:= \mathsf{flow}(c) \cup \{(l, \mathsf{init}(c))\} \cup \\
&\ \{(l', l) \mid l' \in \mathsf{final}(c)\}
\end{aligned}
$$

Representing Control Flow III

Example 18.5

$c = [\texttt{z := 1}]^1;$
$\quad \texttt{while}\ [\texttt{x > 0}]^2\ \texttt{do}$
$\quad\quad [\texttt{z := z*y}]^3;$
$\quad\quad [\texttt{x := x-1}]^4$

Representing Control Flow III

Example 18.5

$$
\begin{aligned}
c = {} & [\texttt{z := 1}]^1; \\
& \quad \texttt{while } [\texttt{x > 0}]^2 \texttt{ do} \\
& \qquad [\texttt{z := z*y}]^3; \\
& \qquad [\texttt{x := x-1}]^4
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{init}(c) &= 1 \\
\mathsf{final}(c) &= \{2\} \\
\mathsf{flow}(c) &= \{(1,2),(2,3),(3,4),(4,2)\}
\end{aligned}
$$

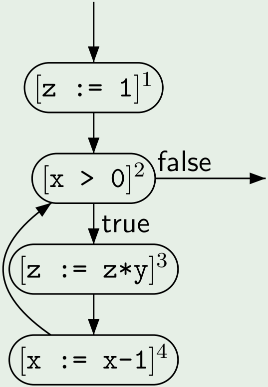
Representing Control Flow III

Example 18.5

Visualization by (control) flow graph:

$$
\begin{array}{l}
c = [\texttt{z := 1}]^1;\\
\quad\texttt{while}\ [\texttt{x > 0}]^2\ \texttt{do}\\
\qquad[\texttt{z := z*y}]^3;\\
\qquad[\texttt{x := x-1}]^4
\end{array}
$$

$$
\begin{aligned}
\mathsf{init}(c) &= 1\\
\mathsf{final}(c) &= \{2\}\\
\mathsf{flow}(c) &= \{(1,2),(2,3),(3,4),(4,2)\}
\end{aligned}
$$



Representing Control Flow IV

- To simplify the presentation we will often assume that the program $c \in Cmd$ under consideration has an isolated entry, meaning that

$$\{l \in L \mid (l, \mathsf{init}(c)) \in \mathsf{flow}(c)\} = \emptyset$$

(which is the case when c does not start with a `while` loop)

Representing Control Flow IV

- To simplify the presentation we will often assume that the program $c \in \mathit{Cmd}$ under consideration has an isolated entry, meaning that

$$\{l \in L \mid (l, \mathsf{init}(c)) \in \mathsf{flow}(c)\} = \emptyset$$

(which is the case when c does not start with a `while` loop)

- Similarly: $c \in \mathit{Cmd}$ has isolated exits if

$$\{l' \in L \mid (l, l') \in \mathsf{flow}(c) \text{ for some } l \in \mathsf{final}(c)\} = \emptyset$$

Representing Control Flow IV

- To simplify the presentation we will often assume that the program $c \in \mathit{Cmd}$ under consideration has an isolated entry, meaning that

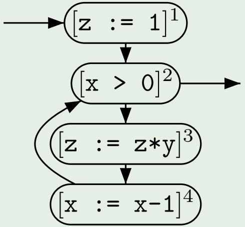
$$\{l \in L \mid (l, \mathsf{init}(c)) \in \mathsf{flow}(c)\} = \emptyset$$

(which is the case when c does not start with a `while` loop)

- Similarly: $c \in \mathit{Cmd}$ has isolated exits if

$$\{l' \in L \mid (l, l') \in \mathsf{flow}(c) \text{ for some } l \in \mathsf{final}(c)\} = \emptyset$$

Example 18.6



has an isolated entry but not isolated exits

Outline

Goal of the Analysis

Available Expressions Analysis

The goal of Available Expressions Analysis is to determine, for each program point, which (complex) expressions *must* have been computed, and not later modified, on all paths to the program point.

Goal of the Analysis

Available Expressions Analysis

The goal of Available Expressions Analysis is to determine, for each program point, which (complex) expressions *must* have been computed, and not later modified, on all paths to the program point.

- can be used to avoid recomputations of expressions
- only interesting for non-trivial (i.e., complex) arithmetic expressions

Goal of the Analysis

Available Expressions Analysis

The goal of Available Expressions Analysis is to determine, for each program point, which (complex) expressions *must* have been computed, and not later modified, on all paths to the program point.

- can be used to avoid recomputations of expressions
- only interesting for non-trivial (i.e., complex) arithmetic expressions

Example 18.7 (Available Expressions Analysis)

```
[x := a+b]^1;
[y := a*b]^2;
while [y > a+b]^3 do
  [a := a+1]^4;
  [x := a+b]^5
```

Goal of the Analysis

Available Expressions Analysis

The goal of Available Expressions Analysis is to determine, for each program point, which (complex) expressions *must* have been computed, and not later modified, on all paths to the program point.

- can be used to avoid recomputations of expressions
- only interesting for non-trivial (i.e., complex) arithmetic expressions

Example 18.7 (Available Expressions Analysis)

$[\texttt{x := a+b}]^1$;
$[\texttt{y := a*b}]^2$;
`while` $[\texttt{y > a+b}]^3$ `do`
  $[\texttt{a := a+1}]^4$;
  $[\texttt{x := a+b}]^5$

- `a+b` available at label 3

Goal of the Analysis

Available Expressions Analysis

The goal of Available Expressions Analysis is to determine, for each program point, which (complex) expressions *must* have been computed, and not later modified, on all paths to the program point.

- can be used to avoid recomputations of expressions
- only interesting for non-trivial (i.e., complex) arithmetic expressions

Example 18.7 (Available Expressions Analysis)

$[\texttt{x := a+b}]^1$;
$[\texttt{y := a*b}]^2$;
`while` $[\texttt{y > a+b}]^3$ `do`
  $[\texttt{a := a+1}]^4$;
  $[\texttt{x := a+b}]^5$

- `a+b` available at label 3
- `a+b` not available at label 5

Goal of the Analysis

Available Expressions Analysis

The goal of Available Expressions Analysis is to determine, for each program point, which (complex) expressions *must* have been computed, and not later modified, on all paths to the program point.

- can be used to avoid recomputations of expressions
- only interesting for non-trivial (i.e., complex) arithmetic expressions

Example 18.7 (Available Expressions Analysis)

$[\texttt{x := a+b}]^1$;
$[\texttt{y := a*b}]^2$;
`while` $[\texttt{y > a+b}]^3$ `do`
  $[\texttt{a := a+1}]^4$;
  $[\texttt{x := a+b}]^5$

- `a+b` available at label 3
- `a+b` not available at label 5
- possible optimization:
  `while` $[\texttt{y > x}]^3$ `do`

Formalizing Available Expressions Analysis I

- Given $c \in Cmd$, $L_c/Block_c/AExp_c$ denote the sets of all labels/blocks/complex arithmetic expressions occurring in c, respectively

Formalizing Available Expressions Analysis I

- Given $c \in Cmd$, $L_c / Block_c / AExp_c$ denote the sets of all labels/blocks/complex arithmetic expressions occurring in c, respectively
- An expression a is killed in a block B if any of the variables in a is modified in B

Formalizing Available Expressions Analysis I

- Given $c \in Cmd$, $L_c / Block_c / AExp_c$ denote the sets of all labels/blocks/complex arithmetic expressions occurring in c, respectively
- An expression a is killed in a block B if any of the variables in a is modified in B
- Formally: $\mathsf{kill}_{\mathsf{AE}} : Block_c \to 2^{AExp_c}$ is defined by

$$\begin{aligned} \mathsf{kill}_{\mathsf{AE}}([\mathtt{skip}]^l) &:= \emptyset \\ \mathsf{kill}_{\mathsf{AE}}([x \ \mathtt{:=}\ a]^l) &:= \{a' \in AExp_c \mid x \in FV(a')\} \\ \mathsf{kill}_{\mathsf{AE}}([b]^l) &:= \emptyset \end{aligned}$$

Formalizing Available Expressions Analysis I

- Given $c \in Cmd$, $L_c/Block_c/AExp_c$ denote the sets of all labels/blocks/complex arithmetic expressions occurring in c, respectively
- An expression a is killed in a block B if any of the variables in a is modified in B
- Formally: $\mathsf{kill}_{\mathsf{AE}} : Block_c \to 2^{AExp_c}$ is defined by

$$\begin{aligned}
\mathsf{kill}_{\mathsf{AE}}([\texttt{skip}]^l) &:= \emptyset \\
\mathsf{kill}_{\mathsf{AE}}([x \texttt{ := } a]^l) &:= \{a' \in AExp_c \mid x \in FV(a')\} \\
\mathsf{kill}_{\mathsf{AE}}([b]^l) &:= \emptyset
\end{aligned}$$

- An expression a is generated in a block B if it is evaluated in and none of its variables are modified by B

Formalizing Available Expressions Analysis I

- Given $c \in Cmd$, $L_c / Block_c / AExp_c$ denote the sets of all labels/blocks/complex arithmetic expressions occurring in c, respectively
- An expression a is killed in a block B if any of the variables in a is modified in B
- Formally: $\mathsf{kill}_{\mathsf{AE}} : Block_c \to 2^{AExp_c}$ is defined by

$$\begin{aligned} \mathsf{kill}_{\mathsf{AE}}([\mathtt{skip}]^l) &:= \emptyset \\ \mathsf{kill}_{\mathsf{AE}}([x \ \mathtt{:=}\ a]^l) &:= \{a' \in AExp_c \mid x \in FV(a')\} \\ \mathsf{kill}_{\mathsf{AE}}([b]^l) &:= \emptyset \end{aligned}$$

- An expression a is generated in a block B if it is evaluated in and none of its variables are modified by B
- Formally: $\mathsf{gen}_{\mathsf{AE}} : Block_c \to 2^{AExp_c}$ is defined by

$$\begin{aligned} \mathsf{gen}_{\mathsf{AE}}([\mathtt{skip}]^l) &:= \emptyset \\ \mathsf{gen}_{\mathsf{AE}}([x \ \mathtt{:=}\ a]^l) &:= \{a \mid x \notin FV(a)\} \\ \mathsf{gen}_{\mathsf{AE}}([b]^l) &:= AExp_b \end{aligned}$$

Example 18.8 ($\mathsf{kill}_{\mathsf{AE}}/\mathsf{gen}_{\mathsf{AE}}$ functions)

$c = [\texttt{x := a+b}]^1;$
$\quad [\texttt{y := a*b}]^2;$
$\quad \texttt{while } [\texttt{y > a+b}]^3 \texttt{ do}$
$\qquad [\texttt{a := a+1}]^4;$
$\qquad [\texttt{x := a+b}]^5$

Example 18.8 ($\mathsf{kill}_{\mathsf{AE}}/\mathsf{gen}_{\mathsf{AE}}$ functions)

$c = [\texttt{x := a+b}]^1;$
$\quad [\texttt{y := a*b}]^2;$
$\quad \texttt{while } [\texttt{y > a+b}]^3 \texttt{ do}$
$\qquad [\texttt{a := a+1}]^4;$
$\qquad [\texttt{x := a+b}]^5$

- $AExp_c = \{\texttt{a+b}, \texttt{a*b}, \texttt{a+1}\}$

Formalizing Available Expressions Analysis II

Example 18.8 ($\mathsf{kill}_{\mathsf{AE}}/\mathsf{gen}_{\mathsf{AE}}$ functions)

c = [x := a+b]^1;
  [y := a*b]^2;
  while [y > a+b]^3 do
    [a := a+1]^4;
    [x := a+b]^5

- $AExp_c = \{\mathtt{a+b}, \mathtt{a*b}, \mathtt{a+1}\}$
-

L_c	$\mathsf{kill}_{\mathsf{AE}}(B^l)$	$\mathsf{gen}_{\mathsf{AE}}(B^l)$
1	$\emptyset$	$\{\mathtt{a+b}\}$
2	$\emptyset$	$\{\mathtt{a*b}\}$
3	$\emptyset$	$\{\mathtt{a+b}\}$
4	$\{\mathtt{a+b}, \mathtt{a*b}, \mathtt{a+1}\}$	$\emptyset$
5	$\emptyset$	$\{\mathtt{a+b}\}$

The Equation System I

- Analysis itself defined by setting up an equation system

The Equation System I

- Analysis itself defined by setting up an equation system
- For each $l \in L_c$, $\mathsf{AE}_l \subseteq AExp_c$ represents the set of available expressions at the entry of block B^l

The Equation System I

- Analysis itself defined by setting up an equation system
- For each $l \in L_c$, $\mathsf{AE}_l \subseteq AExp_c$ represents the set of available expressions at the entry of block B^l
- Formally, for $c \in Cmd$ with isolated entry:

$$\mathsf{AE}_l = \begin{cases} \emptyset & \text{if } l = \mathsf{init}(c) \\ \bigcap\{\varphi_{l'}(\mathsf{AE}_{l'}) \mid (l', l) \in \mathsf{flow}(c)\} & \text{otherwise} \end{cases}$$

  where $\varphi_{l'} : 2^{AExp_c} \to 2^{AExp_c}$ denotes the transfer function of block $B^{l'}$, given by

$$\varphi_{l'}(A) := (A \setminus \mathsf{kill}_{\mathsf{AE}}(B^{l'})) \cup \mathsf{gen}_{\mathsf{AE}}(B^{l'})$$

The Equation System I

- Analysis itself defined by setting up an equation system
- For each $l \in L_c$, $\mathsf{AE}_l \subseteq AExp_c$ represents the set of available expressions at the entry of block B^l
- Formally, for $c \in Cmd$ with isolated entry:
$$\mathsf{AE}_l = \begin{cases} \emptyset & \text{if } l = \mathsf{init}(c) \\ \bigcap\{\varphi_{l'}(\mathsf{AE}_{l'}) \mid (l', l) \in \mathsf{flow}(c)\} & \text{otherwise} \end{cases}$$
where $\varphi_{l'} : 2^{AExp_c} \to 2^{AExp_c}$ denotes the transfer function of block $B^{l'}$, given by
$$\varphi_{l'}(A) := (A \setminus \mathsf{kill}_{\mathsf{AE}}(B^{l'})) \cup \mathsf{gen}_{\mathsf{AE}}(B^{l'})$$
- Characterization of analysis:

  forward: starts in $\mathsf{init}(c)$ and proceeds downwards

  must: $\bigcap$ in equation for AE_l

  flow-sensitive: results depending on order of assignments

The Equation System I

- Analysis itself defined by setting up an equation system
- For each $l \in L_c$, $\mathsf{AE}_l \subseteq AExp_c$ represents the set of available expressions at the entry of block B^l
- Formally, for $c \in Cmd$ with isolated entry:

$$\mathsf{AE}_l = \begin{cases} \emptyset & \text{if } l = \mathsf{init}(c) \\ \bigcap\{\varphi_{l'}(\mathsf{AE}_{l'}) \mid (l', l) \in \mathsf{flow}(c)\} & \text{otherwise} \end{cases}$$

  where $\varphi_{l'} : 2^{AExp_c} \to 2^{AExp_c}$ denotes the transfer function of block $B^{l'}$, given by

$$\varphi_{l'}(A) := (A \setminus \mathsf{kill}_{\mathsf{AE}}(B^{l'})) \cup \mathsf{gen}_{\mathsf{AE}}(B^{l'})$$

- Characterization of analysis:

  forward: starts in $\mathsf{init}(c)$ and proceeds downwards

  must: $\bigcap$ in equation for AE_l

  flow-sensitive: results depending on order of assignments

- Later: solution not necessarily unique

  $\Longrightarrow$ choose greatest one

The Equation System II

Reminder:

$$\mathsf{AE}_l = \begin{cases} \emptyset & \text{if } l = \mathsf{init}(c) \\ \bigcap\{\varphi_{l'}(\mathsf{AE}_{l'}) \mid (l', l) \in \mathsf{flow}(c)\} & \text{otherwise} \end{cases}$$

$$\varphi_{l'}(E) = (E \setminus \mathsf{kill}_{\mathsf{AE}}(B^{l'})) \cup \mathsf{gen}_{\mathsf{AE}}(B^{l'})$$

The Equation System II

Reminder:

$$\mathsf{AE}_l = \begin{cases} \emptyset & \text{if } l = \mathsf{init}(c) \\ \bigcap\{\varphi_{l'}(\mathsf{AE}_{l'}) \mid (l', l) \in \mathsf{flow}(c)\} & \text{otherwise} \end{cases}$$

$$\varphi_{l'}(E) = (E \setminus \mathsf{kill}_{\mathsf{AE}}(B^{l'})) \cup \mathsf{gen}_{\mathsf{AE}}(B^{l'})$$

Example 18.9 (AE equation system)

$c = [$`x := a+b`$]^1$;
$\quad [$`y := a*b`$]^2$;
$\quad$ `while` $[$`y > a+b`$]^3$ `do`
$\qquad [$`a := a+1`$]^4$;
$\qquad [$`x := a+b`$]^5$

The Equation System II

Reminder:

$$\mathsf{AE}_l = \begin{cases} \emptyset & \text{if } l = \mathsf{init}(c) \\ \bigcap\{\varphi_{l'}(\mathsf{AE}_{l'}) \mid (l', l) \in \mathsf{flow}(c)\} & \text{otherwise} \end{cases}$$

$$\varphi_{l'}(E) = (E \setminus \mathsf{kill}_{\mathsf{AE}}(B^{l'})) \cup \mathsf{gen}_{\mathsf{AE}}(B^{l'})$$

Example 18.9 (AE equation system)

$c =$ [x := a+b]^1;
[y := a*b]^2;
while [y > a+b]^3 do
[a := a+1]^4;
[x := a+b]^5

$l \in L_c$	$\mathsf{kill}_{\mathsf{AE}}(B^l)$	$\mathsf{gen}_{\mathsf{AE}}(B^l)$
1	$\emptyset$	{a+b}
2	$\emptyset$	{a*b}
3	$\emptyset$	{a+b}
4	{a+b, a*b, a+1}	$\emptyset$
5	$\emptyset$	{a+b}

The Equation System II

Reminder:

$$\mathsf{AE}_l = \begin{cases} \emptyset & \text{if } l = \mathsf{init}(c) \\ \bigcap\{\varphi_{l'}(\mathsf{AE}_{l'}) \mid (l', l) \in \mathsf{flow}(c)\} & \text{otherwise} \end{cases}$$

$$\varphi_{l'}(E) = (E \setminus \mathsf{kill}_{\mathsf{AE}}(B^{l'})) \cup \mathsf{gen}_{\mathsf{AE}}(B^{l'})$$

Example 18.9 (AE equation system)

$c = [\texttt{x := a+b}]^1;$
$\quad [\texttt{y := a*b}]^2;$
$\quad \texttt{while } [\texttt{y > a+b}]^3 \texttt{ do}$
$\qquad [\texttt{a := a+1}]^4;$
$\qquad [\texttt{x := a+b}]^5$

$l \in L_c$	$\mathsf{kill}_{\mathsf{AE}}(B^l)$	$\mathsf{gen}_{\mathsf{AE}}(B^l)$
1	$\emptyset$	$\{\texttt{a+b}\}$
2	$\emptyset$	$\{\texttt{a*b}\}$
3	$\emptyset$	$\{\texttt{a+b}\}$
4	$\{\texttt{a+b}, \texttt{a*b}, \texttt{a+1}\}$	$\emptyset$
5	$\emptyset$	$\{\texttt{a+b}\}$

Equations:

$$\begin{aligned}
\mathsf{AE}_1 &= \emptyset \\
\mathsf{AE}_2 &= \varphi_1(\mathsf{AE}_1) = \mathsf{AE}_1 \cup \{\texttt{a+b}\} \\
\mathsf{AE}_3 &= \varphi_2(\mathsf{AE}_2) \cap \varphi_5(\mathsf{AE}_5) \\
&= (\mathsf{AE}_2 \cup \{\texttt{a*b}\}) \cap (\mathsf{AE}_5 \cup \{\texttt{a+b}\}) \\
\mathsf{AE}_4 &= \varphi_3(\mathsf{AE}_3) = \mathsf{AE}_3 \cup \{\texttt{a+b}\} \\
\mathsf{AE}_5 &= \varphi_4(\mathsf{AE}_4) = \mathsf{AE}_4 \setminus \{\texttt{a+b}, \texttt{a*b}, \texttt{a+1}\}
\end{aligned}$$

The Equation System II

Reminder:
$$\mathsf{AE}_l = \begin{cases} \emptyset & \text{if } l = \mathsf{init}(c) \\ \bigcap\{\varphi_{l'}(\mathsf{AE}_{l'}) \mid (l', l) \in \mathsf{flow}(c)\} & \text{otherwise} \end{cases}$$
$$\varphi_{l'}(E) = (E \setminus \mathsf{kill}_{\mathsf{AE}}(B^{l'})) \cup \mathsf{gen}_{\mathsf{AE}}(B^{l'})$$

Example 18.9 (AE equation system)

$$\begin{aligned} c = {} & [\mathtt{x\ :=\ a+b}]^1; \\ & [\mathtt{y\ :=\ a*b}]^2; \\ & \mathtt{while}\ [\mathtt{y\ >\ a+b}]^3\ \mathtt{do} \\ & \quad [\mathtt{a\ :=\ a+1}]^4; \\ & \quad [\mathtt{x\ :=\ a+b}]^5 \end{aligned}$$

$l \in L_c$	$\mathsf{kill}_{\mathsf{AE}}(B^l)$	$\mathsf{gen}_{\mathsf{AE}}(B^l)$
1	$\emptyset$	$\{\mathtt{a+b}\}$
2	$\emptyset$	$\{\mathtt{a*b}\}$
3	$\emptyset$	$\{\mathtt{a+b}\}$
4	$\{\mathtt{a+b}, \mathtt{a*b}, \mathtt{a+1}\}$	$\emptyset$
5	$\emptyset$	$\{\mathtt{a+b}\}$

Equations:
$$\begin{aligned} \mathsf{AE}_1 &= \emptyset \\ \mathsf{AE}_2 &= \varphi_1(\mathsf{AE}_1) = \mathsf{AE}_1 \cup \{\mathtt{a+b}\} \\ \mathsf{AE}_3 &= \varphi_2(\mathsf{AE}_2) \cap \varphi_5(\mathsf{AE}_5) \\ &= (\mathsf{AE}_2 \cup \{\mathtt{a*b}\}) \cap (\mathsf{AE}_5 \cup \{\mathtt{a+b}\}) \\ \mathsf{AE}_4 &= \varphi_3(\mathsf{AE}_3) = \mathsf{AE}_3 \cup \{\mathtt{a+b}\} \\ \mathsf{AE}_5 &= \varphi_4(\mathsf{AE}_4) = \mathsf{AE}_4 \setminus \{\mathtt{a+b}, \mathtt{a*b}, \mathtt{a+1}\} \end{aligned}$$

Solution:
$$\begin{aligned} \mathsf{AE}_1 &= \emptyset \\ \mathsf{AE}_2 &= \{\mathtt{a+b}\} \\ \mathsf{AE}_3 &= \{\mathtt{a+b}\} \\ \mathsf{AE}_4 &= \{\mathtt{a+b}\} \\ \mathsf{AE}_5 &= \emptyset \end{aligned}$$